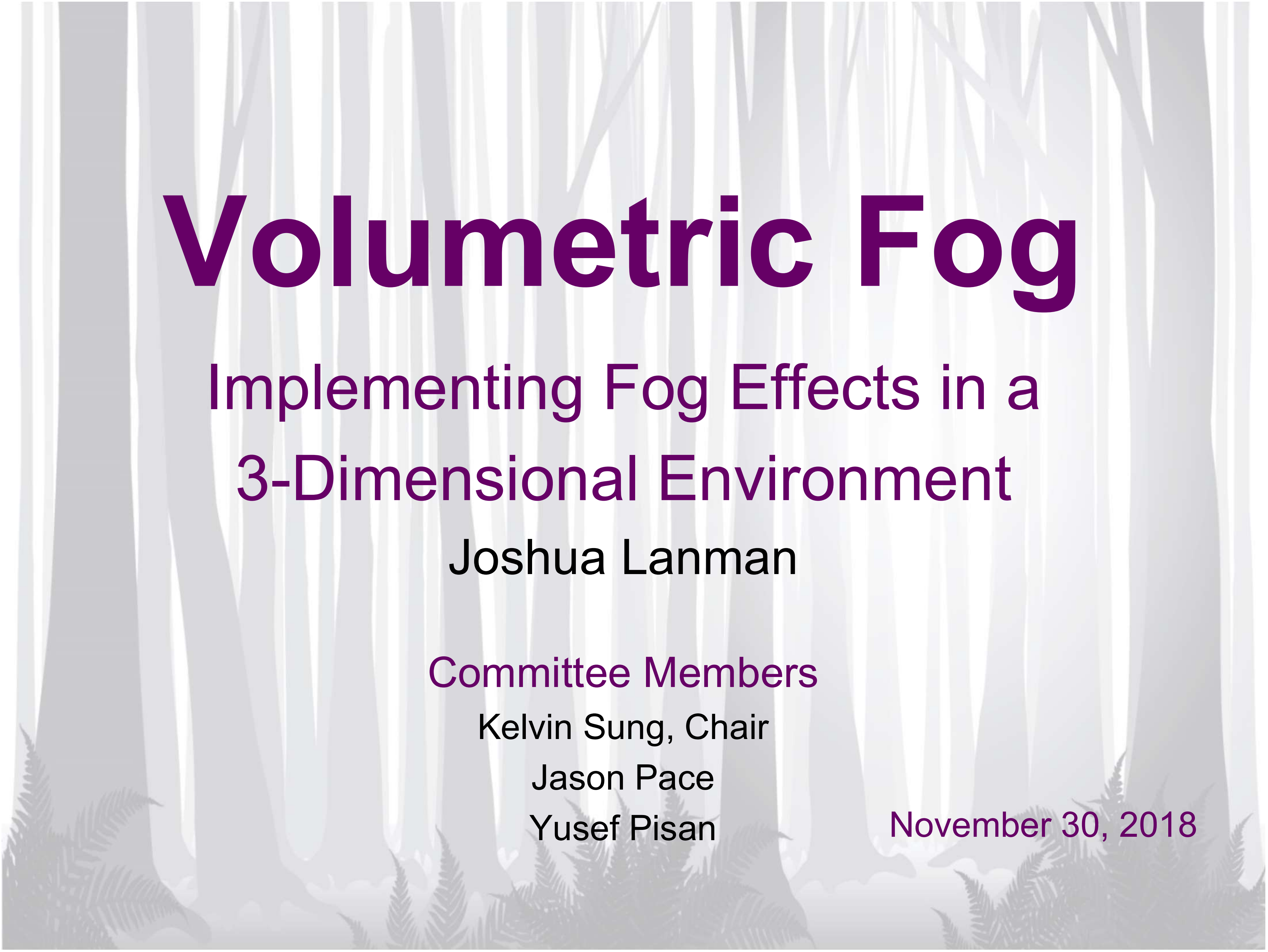


Volumetric Fog

Implementing Fog Effects in a 3-Dimensional Environment

Joshua Lanman

Committee Members

Kelvin Sung, Chair

Jason Pace

Yusef Pisan

November 30, 2018

Outline

- Introduction
- Challenges
- Implementation and Results
- Video Demonstration
- Conclusion/Q&A

Introduction

Introduction

- Since 2000's: Rapid advances in computing

- Volumetric effects
  - Depth
  - Realism
  - Explosions, fire, smoke, rain, fog, dust, shafts of light/shadow, etc.

- Implementation of volumetric effects
  - High-level methodology is widely known
  - Low-level details are not generally available

Introduction

- Focus: Implementing volumetric fog

- Volumetric Fog:
  - Simulates interaction between light and atmosphere
  - Challenging to implement
    - *Mathematically complex*
    - *Computationally expensive*

Introduction

- Three Goals:
    - Understand volumetric fog
    - Build an interactive system
    - Share what we’ve learned

Challenges

Challenges – Illumination Model

- ❖ **Local Illumination**
- ❖ **Global Illumination**
- ❖ **Volumetric Illumination**

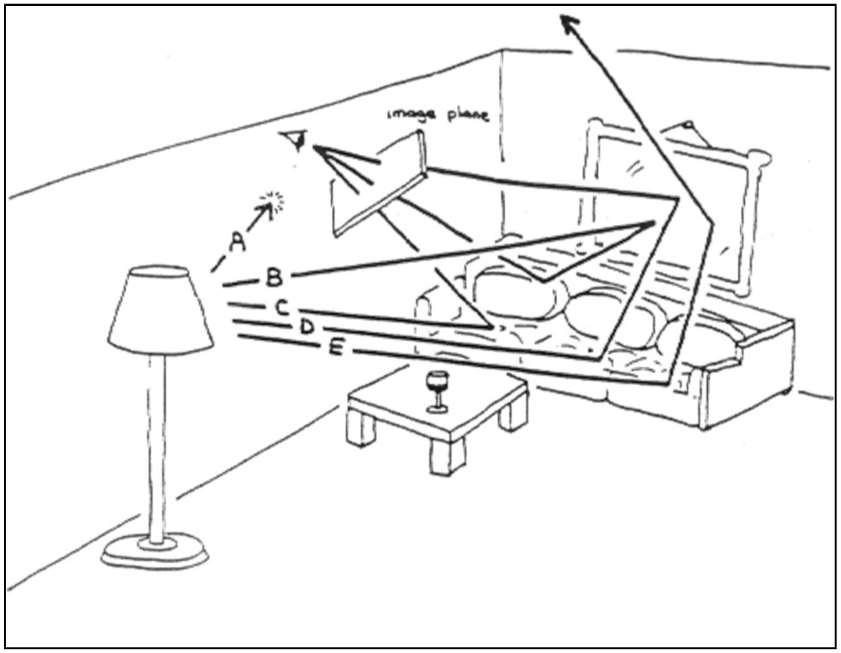


Image Source:
https://www.siggraph.org/education/materials/HyperGraph/illumin/illum1.htm

Sources: [1] – [4]

Challenges – Understanding Fog

- Characteristics of fog
  - Reduced visibility
  - Thickness / height
  - Variations in density
  - Lifting / evaporating
  - Drifting / swirling
  - Variations over time

Sources: [5] – [6]

Five Methods

Method 1: Distance-Based Fog

- Surface shaders
- Blends object color with sky color
- Real-time performance
- Downside: simplicity

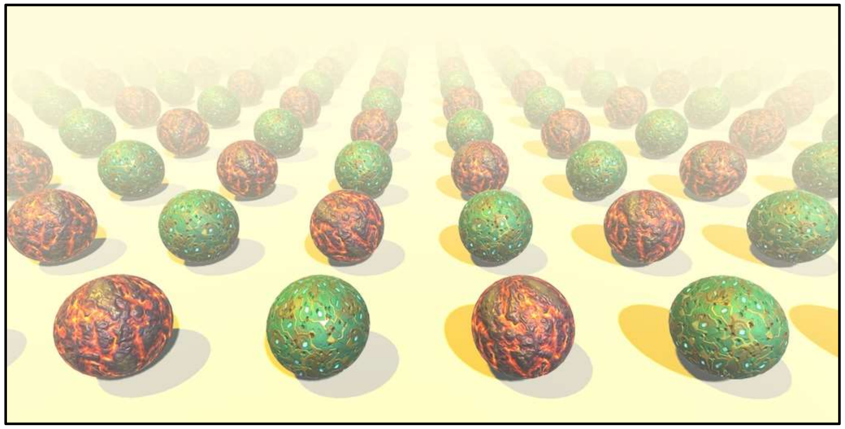


Image Source: https://catlikecoding.com/unity/tutorials/rendering/part-14/

Sources: [7] – [9]

Method 2: Billboarding

- Particle effects on a 2D surface
- Always facing the camera
- Supports local effects
- Downsides:
  - One viewing angle
  - Hard edges

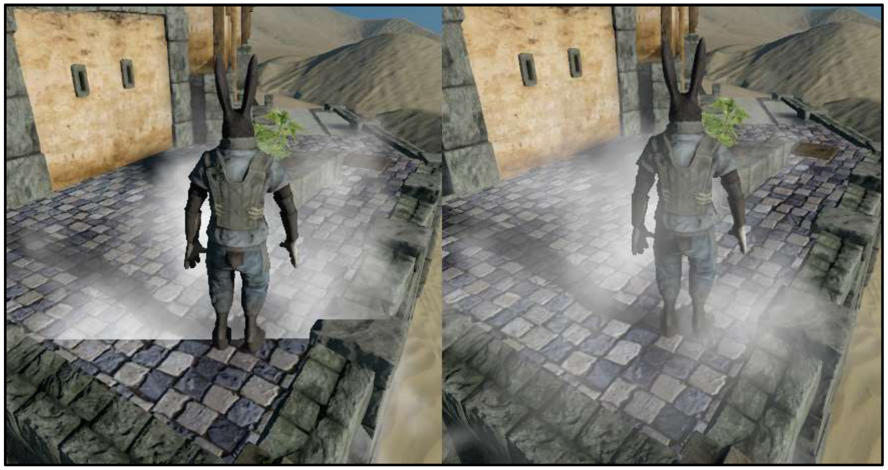


Image Source: http://blog.wolfire.com/2010/04/Soft-Particles

Sources: [7], [10]

Method 3: Particle Emission

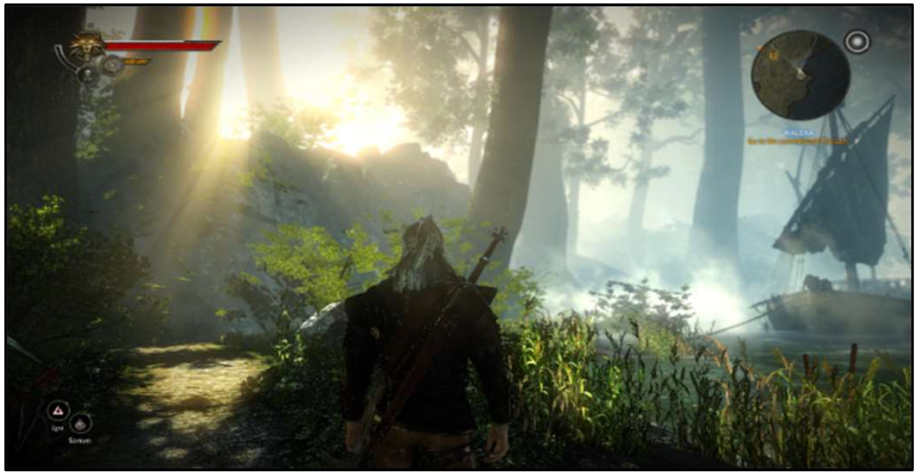
- ❖ Many small particles
- ❖ Good visual quality
- ❖ Highly dynamic effects
- ❖ Downside:
  - ❖ Overhead

Image Source: https://www.youtube.com/watch?v=lekE0Ez_go0

Sources: [7], [11] – [13]

Method 4: Post-Effect Image-Based

- Popularized by Unreal Engine 3.0 & CryEngine
- Uses photography post-processing techniques
- Excellent results under the right conditions
- Weaknesses:
  - Not physics based
  - No support for close lights
  - Fog / transparent objects
  - Directionality



Sources: [7]

Image Source: https://www.gamasutra.com/blogs/BartlomiejWronski/20141208/226295/Atmospheric_scattering_and_volumetric_fog_algorithm__part_1.php

Method 5: Volumetric Fog

- Current technique
- Post-effect
  - After depths, normal, and colors calculated
- GPU: per-pixel calculations
  - Raymarching

Sources: [7]

Image Source: https://gkan.artstation.com/projects/o1bRO

Method 5: Volumetric Fog

- Photo-realistic
- Simulate real world
- Support multiple lights
- Current implementations
  - Several tradeoffs
  - Some limitations

Sources: [7]

Image Source:
https://gkan.artstation.com/projects/o1bRO

Summary

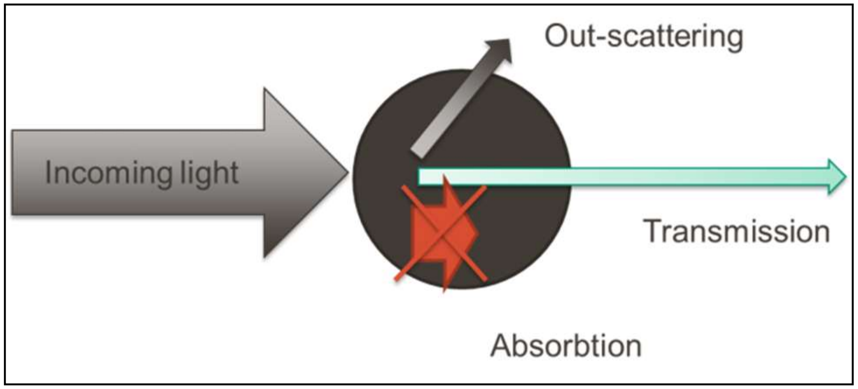
- Five methods for generating fog
  - Merits / trade offs

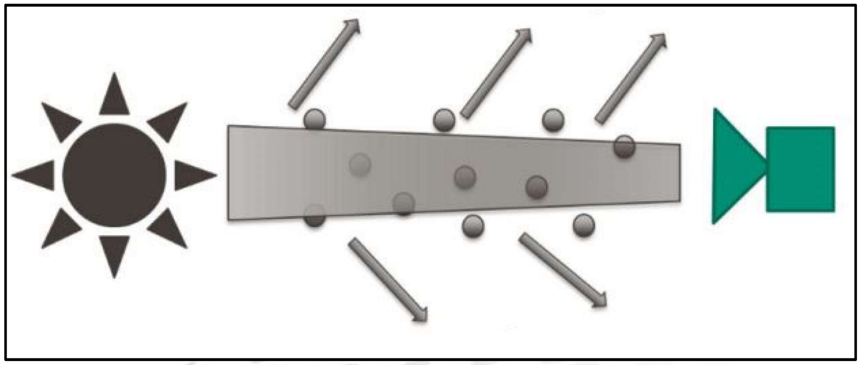
- Our focus: Volumetric approach to simulating fog
  - Most recently developed
  - Follows and approximates real-life physics
  - Results in visually appealing / realistic fog

Volumetric Considerations

Implementation – Considerations

- Atmospheric Absorption and Scattering of Light
  - Light-Particle Interaction
  - Occurs over entire distance traveled
  - Beer-Lambert Law





Sources: [14] – [17]

Image Source: https://bartwronski.files.wordpress.com/2014/08/bwronski_volumetric_fog_siggraph2014.pdf

Implementation – Considerations

Beer-Lambert Law

$$I = I_0 * e^{-\beta e * x}$$

Where

I_0 – Initial light intensity

x – Distance traveled through the participating medium

βe – (the extinction coefficient) Sum of the scattering and absorption coefficients

If βe is a function of x, then the equation becomes:

$$I_{(A\rightarrow B)} = I_0 * e^{-\int_A^B \beta e(x)\, dx}$$

Sources: [14] – [17]

Implementation – Considerations

Beer-Lambert Law (Our Implementation)

$$I_{pixel} = I_0 * \sum_{i=0}^{n} F_1(x, y, z) * \cdots * F_j(x, y, z)$$

Where

I_0 – Initial light intensity

$F_j(x,y,z)$ – Attenuation factors (for each feature implemented)

Implementation – Considerations

- At each raymarching step (1 to n), we determine the density and color of the fog
  - Return these values for every pixel and frame rendered

- The final color of each pixel is determined by the equation:

$$\boldsymbol{PixelColor = Color * (1 - d) + fogColor * d}$$

Where

color – original color value for the pixel

d – total fog density for the pixel

Implementation And Results

Base Scene

- ❖ **Here is a simple forest scene with mountains…**

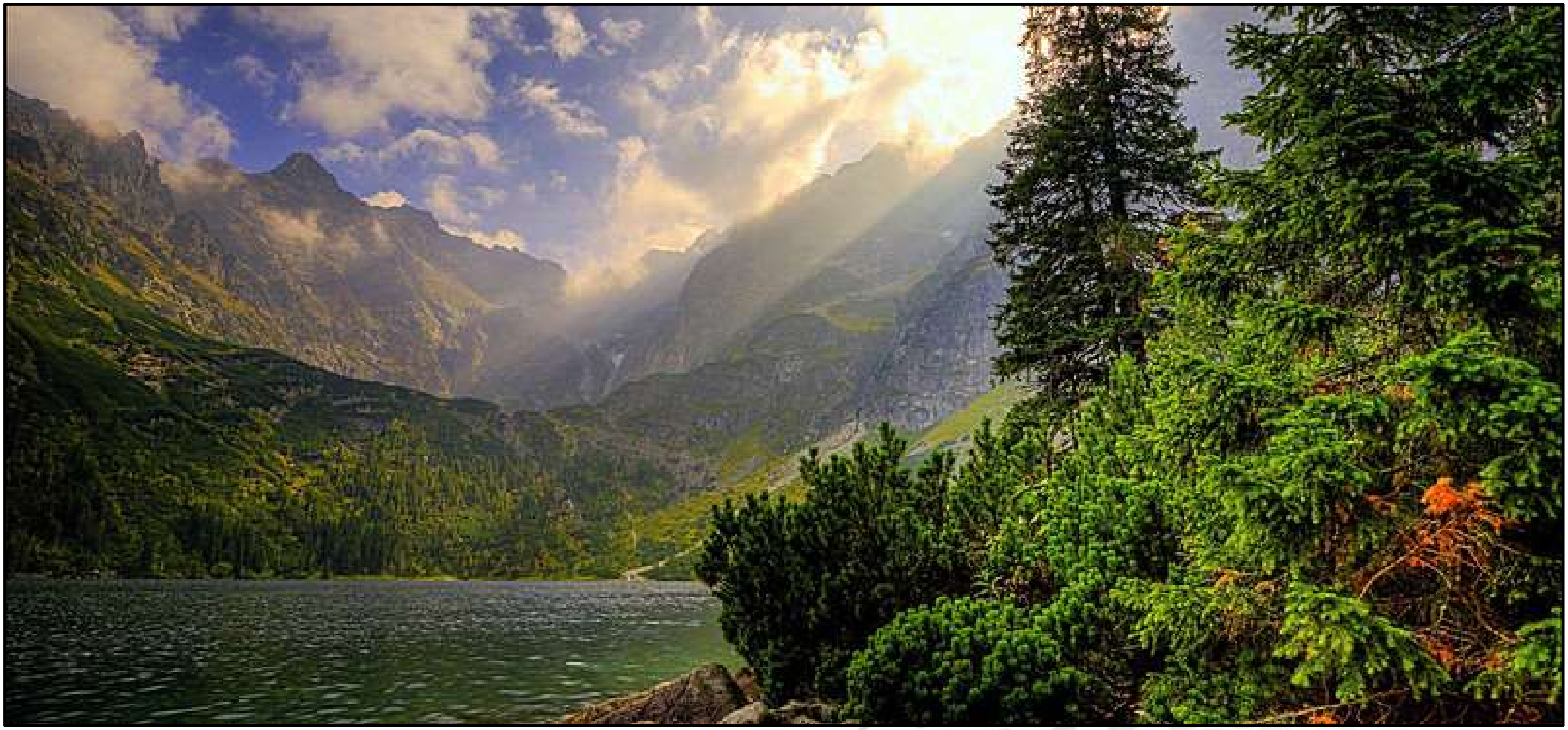
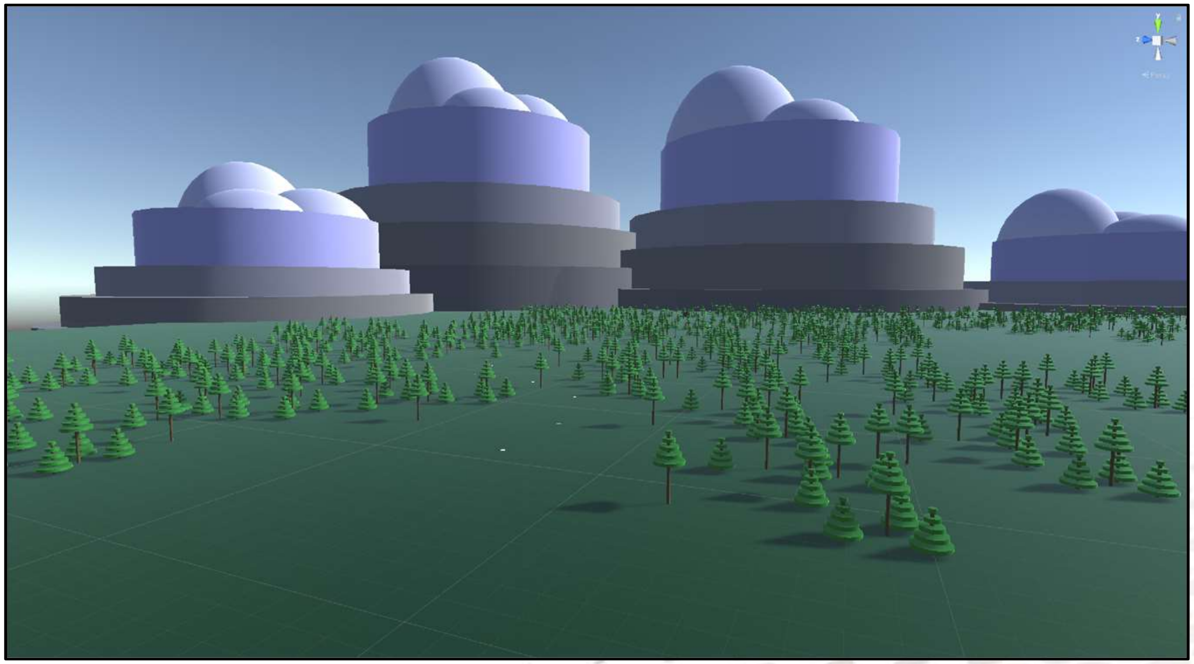


Image Source:
https://pngtree.com/freebackground/mountain-landscape-range-forest_619115.html

Base Scene

- Hundreds of 3D primitives
  - Ground plane
  - Mountains
  - Trees (two sizes)
  - Shed and lamp post
  - Cookie trail



Density Attenuation Factors

- Factor 1: Constant depth fog
- Factor 2: Height density
- Factor 3: Edge density
- Factor 4: Variable-density

Fog Density – Implementation

- Basic user input:
  - Number of steps to use in raymarching (from 2 to 256)
  - Depth at which the fog becomes to dense to see through
  - Scene-wide fog or localized (placed in a box in the scene)
- From the scene:
  - Distance to the fragment (linear value from 0 to 1)
- Using this information, we calculate:
  - Size of each step
  - Density of the fog to accumulate each step

Depth Fog – Implementation

- **<u>Goal</u>: Constant density fog**
- **<u>Desired Result</u>: Fog that uniformly obscures objects with distance**
- **Uniform density for each raymarching step**
- **Final density: sum of fog values calculated at each step**

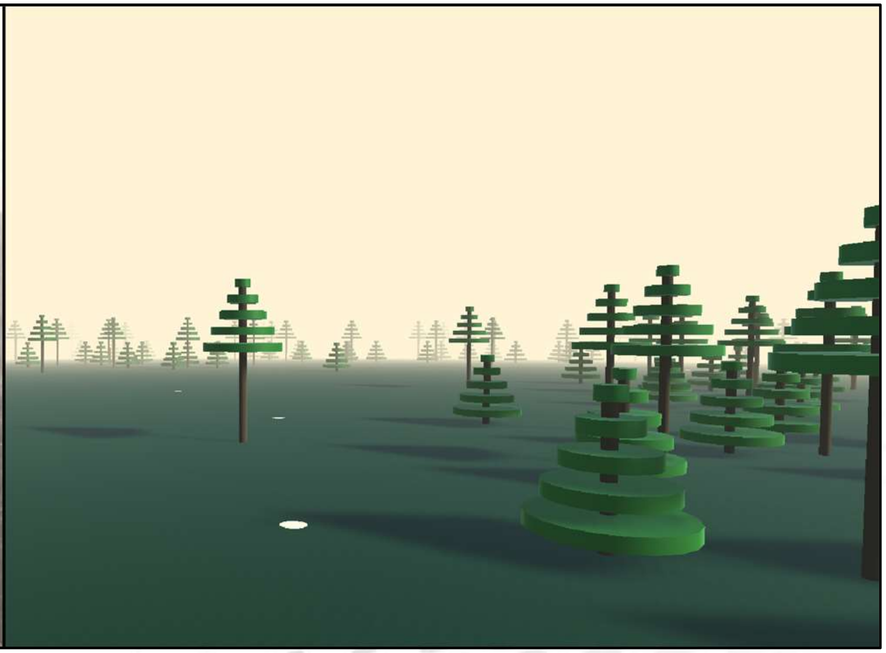
$$I_{pixel} = I_0 * \sum_{i=0}^{n} F_{depth}(x, y, z)$$

Image Source: https://www.videoblocks.com/video/working-tractor-in-the-distance-during-the-fog-rvdxkvbggivx9p4pv

Depth Fog – Results

- ❖ Close objects appear normal
- ❖ Distant objects appear faded and indistinct
- ❖ Most distant objects are completely obscured

Real World



Generated

Height Density – Implementation

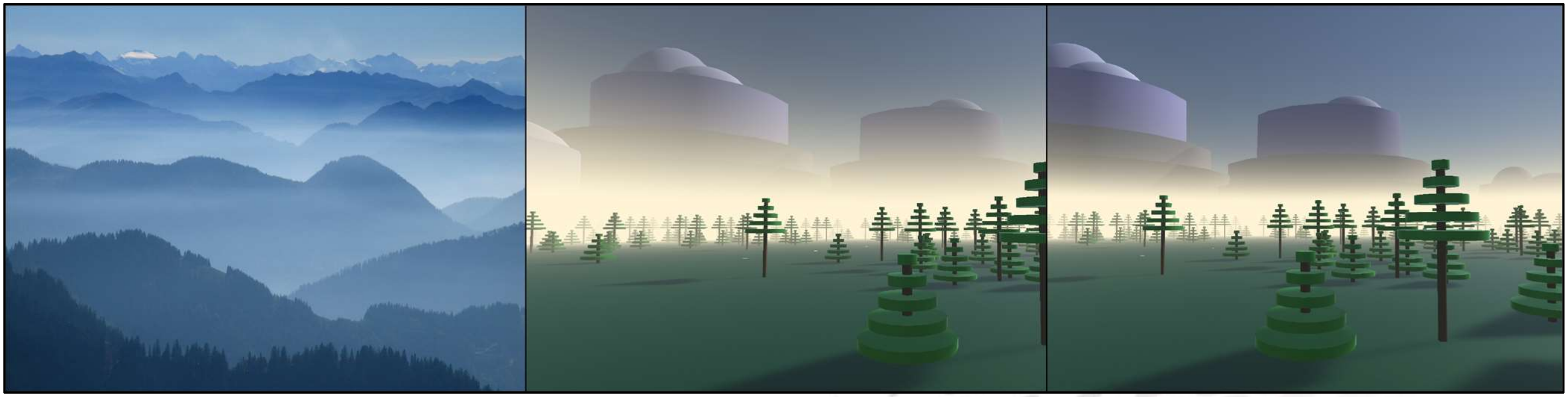
- Observation: Density varies with altitude.
  - Thicker at bottom
  - Thins at the top
- User provides three values:
  - Height
  - Distance
  - Linear or exponential

$$I_{pixel} = I_0 * \sum_{i=0}^{n} F_{depth} * F_{height}$$

Image Source: https://naturfotografen-forum.de/o21898-Blaue%20Stunde%20ND

Height Density – Results

- Two functions shown:
  - Linear (center)
  - Exponential (right)
- Bottom: Constant density fog
- Top: Varies with altitude



Real World

Linear

Exponential

Edge Density – Implementation

- Observation: Density varies at the fog edge.
- Similar mechanism to the height density
- User provides two values per direction
  - Distance
  - Linear or exponential

$$I_{pixel} = I_0 * \sum_{i=0}^{n} F_{depth} * F_{height} * F_{edge}$$

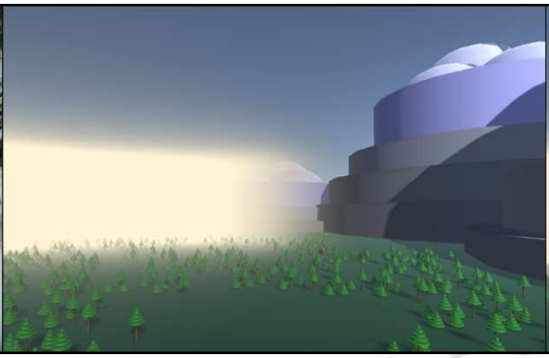
Image Source:
https://www.videoblocks.com/video/working-tractor-in-the-distance-during-the-fog-rvdxkvbggivx9p4pv

Edge Density – Results

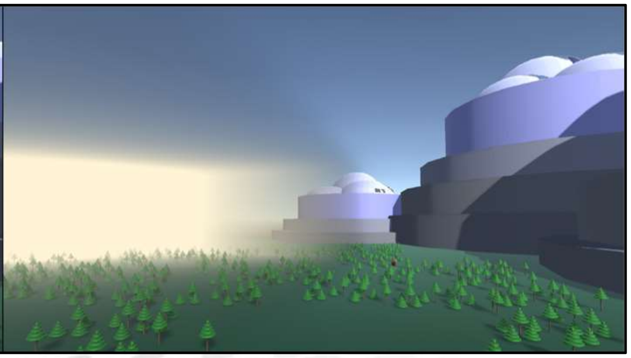
- Two functions shown:
  - Linear (center)
  - Exponential (right)

- Difference between the two is subtle
  - Exponential function falls off over a shorter distance

Real World



Linear



Exponential

Variable-Density – Implementation

- Observation: Density varies as position and time change.
- Changes in visibility
- Observe fog motion

Image Source:
http://wikilovesearth.org/wle-2015-cloudy-and-foggy-nature/

Variable-Density – Implementation

- Simplex noise function
  - HLSL implementation
  - Pseudorandom in 2D/3D/4D
  - Simple function call
  - Values in range 0 to 1

- User provides:
  - Noise strength

Image Source:
http://wikilovesearth.org/wle-2015-cloudy-and-foggy-nature/

Variable-Density – Implementation

- Step density is modified by a factor of the noise strength times the random noise value

$$I_{pixel} = I_0 * \sum_{i=0}^{n} F_{depth} * F_{height} * F_{edge} * F_{random}$$

Image Source: http://wikilovesearth.org/wle-2015-cloudy-and-foggy-nature/

Drifting Fog – Implementation

- Builds on variable-density
- User provides:
  - Velocity vector
- Velocity * _Time.x

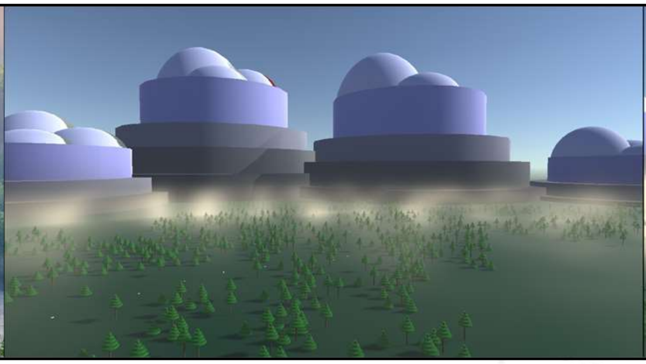
$$I_{pixel} = I_0 * \sum_{i=0}^{n} F_{depth} * F_{height} * F_{edge} * F_{random}$$

Image Source:
http://wikilovesearth.org/wle-2015-cloudy-and-foggy-nature/

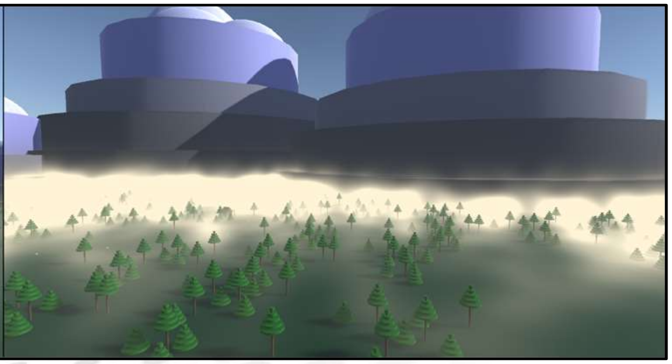
Variable-Density – Results

- ❖ Two different scenarios shown below:
  - ❖ Thin, wispy ground fog (center)
  - ❖ Thicker, rolling fog (right)
- ❖ Ground fog: Patchy, cloud-like appearance
- ❖ Rolling fog: Objects fade in and out with drift

Real World



Thin Ground Fog



Thicker Rolling Fog

Fog Color

- Base color
- Shadows
- Ambient fog
- Multiple lights

Fog Color – Implementation

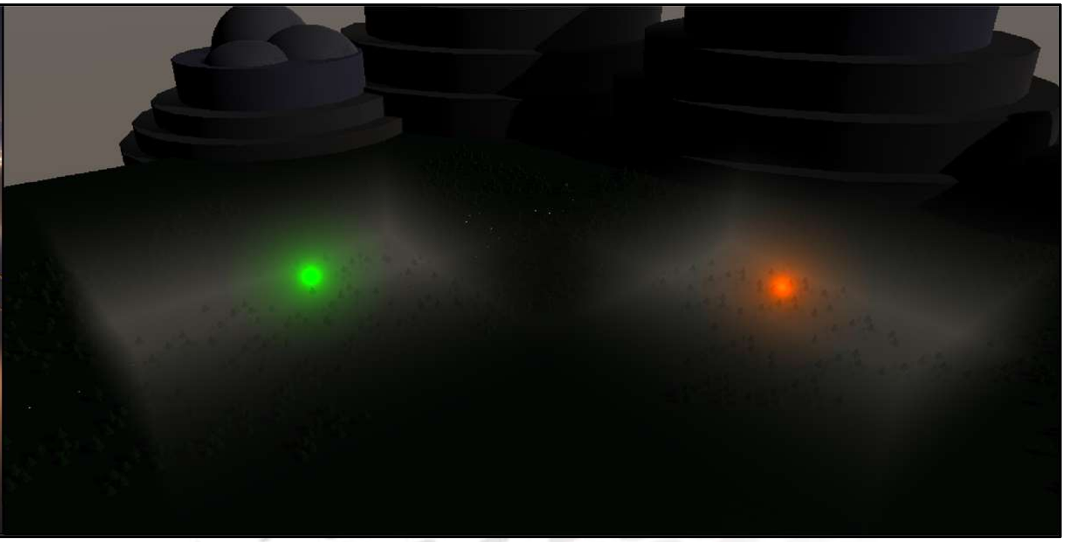
- Real world: Color of fog based on light interactions
- Initially: One light
- Fog color is light color multiplied by step density
- Results accumulated for each pixel and blended with original pixel color

Image Source:
http://www.shainblumphoto.com/project/symphony-of-the-fog/

Fog Color – Results

- ❖ Fog and game objects:
  - ❖ Both directly affected by scene lighting

Real World



Generated

Shadows – Implementation

- High dynamic lighting
  - Crepuscular rays
- Shadow maps
  - Passed to shader using a command buffer
  - Get shadow status by location
  - User: shadow strength
- Multiply the lit fog color by both shadow strength and shadow status

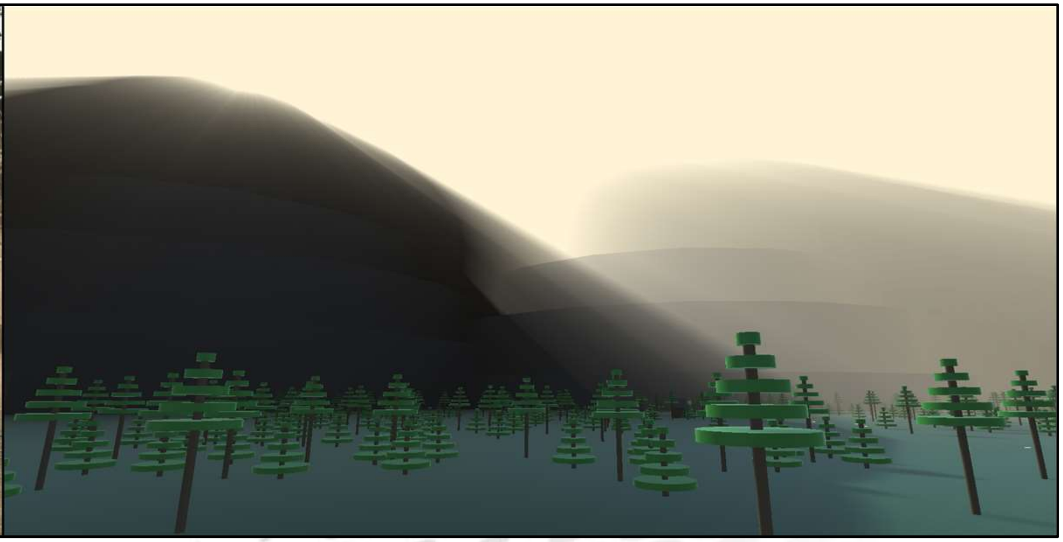
Image Source: https://astrobob.areavoices.com/2011/07/10/crepuscular-rays-a-tale-of-sunbeams-diverging-in-the-blue-sky/

Shadows – Results

- **Partial success:**
  - **Generated fog displays shafts of light and shadow**

- **Future improvements:**
  - **Scattering functions (Raleigh, Mie, others)**

Real World



Generated

Ambient Fog – Implementation

- Not natural occurrence
- Useful effect in movies and games
- User provides two values:
  - Intensity
  - Color
- Linear interpolation
  - Step color
  - Ambient color

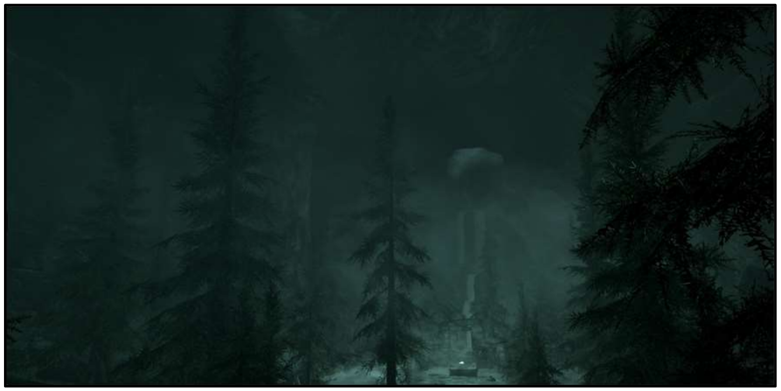
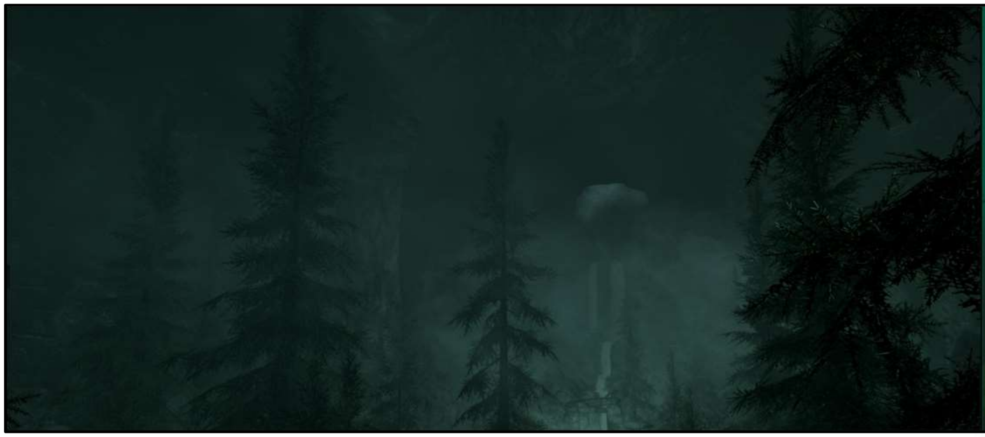


Image Source: https://www.nexusmods.com/skyrimspecialedition/mods/11197/

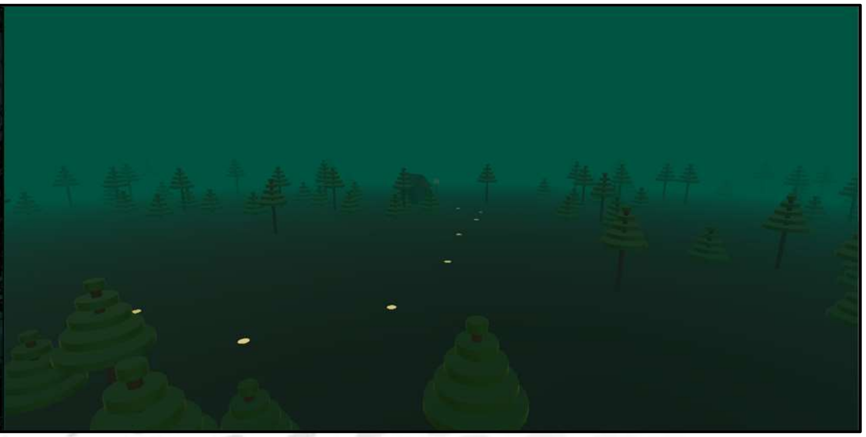
Ambient Fog – Results

- Effect we are replicating (Skyrim)
    - Glowing blue-green fog in a cave with low lighting

- Directional light disabled (no scene lighting)
    - Glowing green fog
    - Useful in lit fog: Can modify fog color



Borrowed from the game, *Skyrim*



Generated

Point Light – Implementation

- Inputs from light source:
  - Location
  - Range
  - Intensity
  - Color
- Three attenuation factors from the user:
  - Constant
  - Linear
  - Exponential

Image Source: http://www.romeofthewest.com/2009/01/dense-fog.html

Point Light – Implementation

- **Process:**

1. **Direction to point light**
2. **Distance to point light**

If in point light's range:

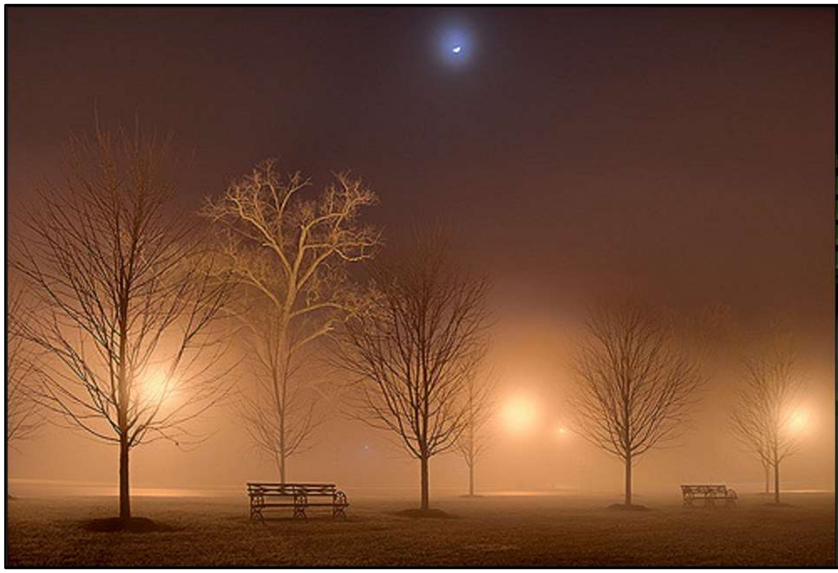
3. **Point light intensity at the current location**
4. **Linearly interpolate between step color and point light color**

Image Source:
http://www.romeofthewest.com/2009/01/dense-fog.html

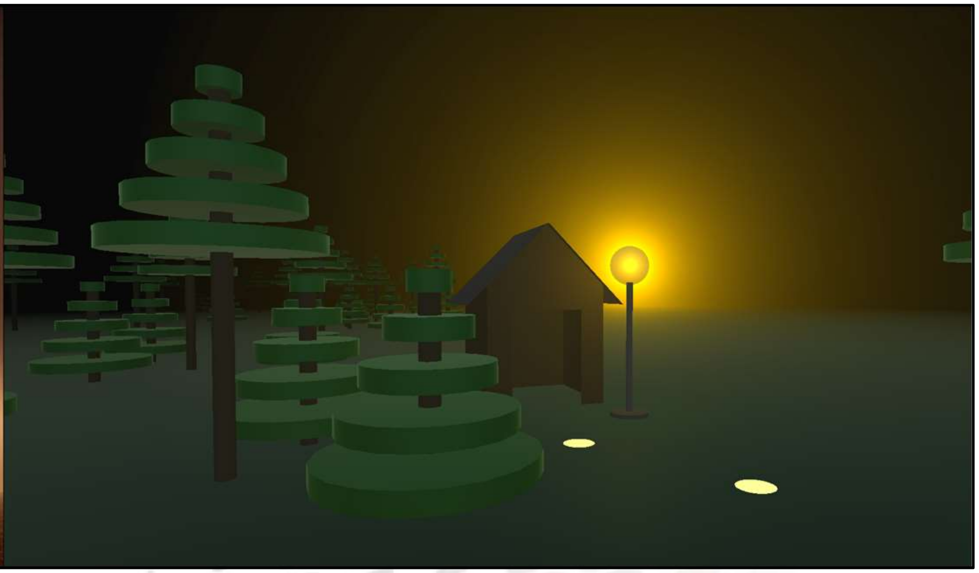
Multiple Lights/Point Light – Results

- ❖ Visual quality can approximate the real world

- ❖ Result does not exactly match reality
  - ❖ Extra light attenuation factors – *Artistic endeavor*



Real World



Generated

Results – Performance

- Goal
  - Demonstrate a highly performant fog simulation
  - Running at a full HD resolution of 1920 x 1080
  - Game scene of moderate complexity
- Results
  - Achieved our performance goals (see video)
- Analysis
  - Largest performance detractor: variable density/noise
  - Variable density can decrease frame rates by up to ~75%

Video Demonstration

Conclusion

- Researched and implemented a system for generating volumetric fog
  - General process is widely known, however…
  - Optimizations generally are not known (trade secrets)

Conclusion

- ❖ Future Improvements
  - ❖ Model: Linear method of stacking effects may impact ability to deliver advanced features (reality: fog aspects are interrelated)
  - ❖ Model: Add support for multiple overlapping fogs.
  - ❖ Optimization: Explore optimizations for variable-density fog (noise function)

Conclusion

- **Future Improvements (Continued)**
  - **<u>Feature</u>: Need support for additional light types, such as spots, and more lights in the scene**
  - **<u>Feature</u>: Improved shadows in fog (shadow cascades / perspective aliasing)**
  - **<u>Usability</u>: Interface**

Conclusion

- Good progress on many aspects / features of fog
  - Entire scene or localized fog
  - Linear / exponential height density
  - Edge density for localized fog
  - Shadowed fog
  - Noisy / variable-density fog
  - Others…

Conclusion

- Available (by quarter's end) for researcher's and enthusiasts to use as a starting point in their own projects
  - UW Bothell's Digital Future Lab (DFL) has shown interest

Questions?

Sources

[1] “GPU Gems - Chapter 39. Volume Rendering Techniques.” [Online]. Available: http://developer.download.nvidia.com/books/HTML/gpugems/gpugems_ch39.html. [Accessed: 10-Nov-2018].

[2] D. Zorin, “Lighting,” *New York University*, 2005. [Online]. Available: https://mrl.nyu.edu/~dzorin/cg05/lecture08.pdf. [Accessed: 28-Oct-2018].

[3] Singh, Karan, “CSC418 Computer Graphics: Illumination,” Fall-2007. [Online]. Available: http://www.dgp.toronto.edu/~karan/courses/csc418/lectures/l15.pdf. [Accessed: 09-Nov-2018].

[4] Owen, G. Scott, “Illumination Models - Introduction,” *Illumination Models*. [Online]. Available: https://www.siggraph.org/education/materials/HyperGraph/illumin/illum1.htm. [Accessed: 10-Nov-2018].

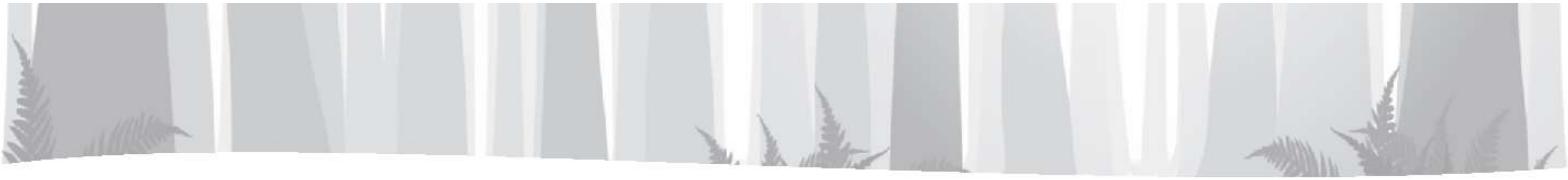
[5] Perez-Diaz, Jose L., Ivanov, Ognyan, Peshev, Zahary, and Alvarez-Valenzuela, Marco A., “Fogs: Physical Basis, Characteristic Properties, and Impacts on the Environment and Human Health,” *ResearchGate*, 20-Oct-2017. [Online]. Available: https://www.researchgate.net/publication/320531889_Fogs_Physical_Basis_Characteristic_Properties_and_Impacts_on_the_Environment_and_Human_Health. [Accessed: 11-Nov-2018].

[6] Croft, P J and University of Louisiana at Monroe, “Fog,” *Fog*, 2003. [Online]. Available: http://curry.eas.gatech.edu/Courses/6140/ency/Chapter8/Ency_Atmos/Fog.pdf. [Accessed: 10-Nov-2018].

[7] “Gamasutra: Bartlomiej Wronski’s Blog - Atmospheric scattering and ‘volumetric fog’ algorithm – part 1.” [Online]. Available: https://www.gamasutra.com/blogs/BartlomiejWronski/20141208/226295/Atmospheric_scattering_and_volumetric_fog_algorithm__part_1.php. [Accessed: 15-Jul-2018].

[8] J. Mackay, “In Praise of Video Gaming’s Old Dalliance with Distance Fog,” *Waypoint*, 07-Feb-2017. [Online]. Available: https://waypoint.vice.com/en_us/article/mg9p3a/in-praise-of-video-gamings-old-dalliance-with-distance-fog. [Accessed: 29-Sep-2018].

[9] Jasper Flick, “Rendering 14, Fog, a Unity Tutorial.” [Online]. Available: https://catlikecoding.com/unity/tutorials/rendering/part-14/. [Accessed: 30-Jun-2018].



Sources

[10] “Soft Particles - Wolfire Games Blog.” [Online]. Available: http://blog.wolfire.com/2010/04/Soft-Particles. [Accessed: 30-Jun-2018].

[11] “Mist System - Highly efficient flowing mist, fog or dust - Unity Forum.” [Online]. Available: https://forum.unity.com/threads/mist-system-highly-efficient-flowing-mist-fog-or-dust.269187/. [Accessed: 29-Sep-2018].

[12] A. Zanjiran, “[Tutorial] Volumetric Fog in Unity 2017 using New Particle Shader,” *YouTube*, 03-Jan-2018. [Online]. Available: https://www.youtube.com/watch?v=STugI38kD8c. [Accessed: 30-Jun-2018].

[13] SpeedTutor, “[Unity3D] Creating a mist / fog particle effect with shuriken,” *YouTube*, 26-Jun-2014. [Online]. Available: https://www.youtube.com/watch?v=lekE0Ez_go0. [Accessed: 30-Jun-2018].

[14] W. Jarosz, “Efficient Monte Carlo Methods for Light Transport in Scattering Media,” *Dartmouth University*, Sep-2008. [Online]. Available: https://cs.dartmouth.edu/~wjarosz/publications/dissertation/chapter4.pdf. [Accessed: 01-Jul-2018].

[15] Wronski, Bartlomiej, “Volumetric Fog: Unified Compute Shader Based Solution to Atmospheric Scattering (Presented at Siggraph 2014),” 2014. [Online]. Available: https://bartwronski.files.wordpress.com/2014/08/bwronski_volumetric_fog_siggraph2014.pdf.

[16] Carn, Dr. Simon A., “Fundamentals of Remote Sensing - Atmospheric Transmission,” *Michigan Technological University*, 31-Mar-2010. [Online]. Available: http://pages.mtu.edu/~scarn/teaching/GE4250/transmission_lecture.pdf. [Accessed: 11-Nov-2018].

[17] Prof. Glenn H. Chapman: SFU Eng. Science, “Lesson 7: Absorption & Scattering,” *ENSC 376: Introduction to Optical Engineering and Design*, 30-Jan-2008. [Online]. Available: http://www2.ensc.sfu.ca/~glennc/e376/e376l7.pdf. [Accessed: 01-Oct-2018].